IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the Matter of the Application of: John Michael Lake
Serial No.: 10/801,369 Confirmation No.: 3169
Filed: March 16, 2004
For: Determining Software Complexity
Examiner: Anil Khatri Group Art Unit: 2163
Attorney Docket No.: RSW920040039US1

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

In response to the Office Action of February 6, 2008, and in support of the Notice of Appeal file on March 22, 2008, Applicants respectfully submit this Appeal Brief.

(I) Real Party in Interest

The real party in interest for this Application is assignee INTERNATIONAL BUSINESS MACHINES CORPORATION of Armonk, NY.

(II). Related Appeals and Interferences

There are no related appeals or interferences.

(III). Status of Claims

Claims 9-19 are pending in this case. Claims 9-19 stand rejected under 35 USC 102 as being anticipated by US Patent Number 6,496,974 to Sliger et al. Claims 1-8 have been canceled. Claims 9-19 are being appealed.

(IV). Status of Amendments

No amendments have been made to the claims.

(V). Summary of claimed subject matter

(V.A) Claim 9

Claim 9 is directed to an apparatus for determining complexity of a software component (Fig. 2 and paragraphs 001, 006, 008, 009, 018). The apparatus comprises logic (200 in Fig. 2, paragraph 018 and paragraph 019) for determining a plurality of versions of the software component (P0, P1, P2; Steps 100, 110, 120 in Fig. 1, paragraphs 010, 013, 014, 019) and for finding lengths of compressed versions of the plurality of versions of the software (C0, C1, C2, Steps140, 150, 160 in Fig. 1, paragraphs 010, 016, 019). The apparatus also comprises means for compressing each of the versions (210 in Fig. 2, paragraphs 018, 019) to provide the compressed versions, and means for

comparing the lengths of the compressed versions (220 in Fig. 2, Step 170 in Fig. 1, paragraph 018 and paragraph 019). A display, other output device or equivalent provides a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraphs 006, 017, 020).

(V.B) Claim 10

Claim 10 is directed to an apparatus for determining complexity of a software component (Fig. 2 and paragraphs 001, 006, 008, 009, 018). The apparatus comprises logic (200 in Fig. 2, paragraphs 018, 019) for creating raw program text (P0; Step 100 in Fig. 1, paragraphs 013, 014, 019) and normalized program text (P1; step 110 in Fig. 1, paragraphs 013, 014, 019) of the software component and for finding lengths of compressed raw program text (C0; Step 140 in Fig. 1, paragraphs 016, 019) and compressed normalized program text (C1; Steps 150 in Fig. 1, paragraphs 016, 019). The apparatus also comprises means for compressing the raw program text and the normalized program text (210 in Fig. 2, paragraphs 018, 019) to provide the compressed versions (Step 1130 in Fig. 1, paragraphs 018), and means for finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text (220 in Fig. 2, Step 170 in Fig. 1, paragraph 018 and paragraph 019).

(V.C) Claim 11

Claim 11 is directed to an apparatus for determining complexity of a software component (Fig. 2 and paragraphs 001, 006, 008, 009, 018). The apparatus comprises logic (200 in Fig. 2, paragraphs 018, 019) for creating normalized program text (P1; step 110 in Fig. 1, paragraphs 013, 014, 019) and normalized unique program text (P2; Step120 in Fig. 1, paragraphs 013, 014, 019) of the software component and for finding lengths of compressed normalized program text (C1; Steps 150 in Fig. 1, paragraphs 016, 019) and compressed normalized unique program text (C2; Step 160 in Fig. 1, paragraphs 016, 019). The apparatus also comprises means for compressing the normalized program text and the normalized unique program text (210 in Fig. 2, paragraphs 018, 019) to

provide the compressed versions (Step 130 in Fig. 1, paragraphs 018), and means for finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text (C1/C2; 220 in Fig. 2, Step 170 in Fig. 1, paragraph 018 and paragraph 019). A means for providing a software complexity metric comprising the ratio of the lengths of the compressed versions is provided by a display or other output device or equivalent (paragraphs oo0, 017, 020).

(V.D) Claim 12

Claim 12 is directed to a program storage device (paragraphs 001, 006, 009, 020, 021) readable by machine (paragraph 020), tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component (paragraph 020). The program of instruction creates a plurality of versions of the software component (P0, P1, P2; Steps 100, 110, 120 in Fig. 1, paragraphs 010, 013, 014, 019). The program of instruction compresses each of the versions, to provide compressed versions (C0, C1, C2; Step 130 in Fig. 1, paragraphs 006, 010, 015, 019). The program of instruction finds lengths of the compressed versions (Steps 140, 150, 160 in Fig. 1, paragraphs 016, 019). The program of instruction compares the lengths of the compressed versions (Step 170 in Fig. 1, paragraphs 006, 010, 016, 017, 019). The program of instructions provides a software complexity metric comprising a comparison of the lengths of the compressed versions is provided by a display or other output device or equivalent (paragraphs oo0, 017, 020).

(V.F) Claim 13

Claim 13, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the plurality of versions comprises raw program text (PO; Step 100 in Fig. 1, paragraphs 013, 014, 019).

(V.F) Claim 14

Claim 14, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the plurality of versions comprises normalized program text (P1; Step 110 in Fig. 1, paragraphs 013, 014, 019).

(V.G) Claim 15

Claim 15, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the plurality of versions comprises normalized unique program text (P2; Step 120 in Fig. 1, paragraphs 013, 014, 019).

(V.H) Claim 16

Claim 16, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein the comparison of compressed versions comprises finding a ratio using the length of the compressed version of raw program text and the length of the compressed version of normalized program text (C0/C1; Step 170 in Fig. 1, paragraph 018 and paragraph 019).

(V.I) Claim 17

Claim 17, which depends from claim 12 is directed to a program storage device according to its parent claims and wherein wherein the comparison of compressed versions comprises finding a ratio using the length of the compressed version of normalized program text and the length of the compressed version of normalized unique program text C1/C2; Step 170 in Fig. 1, paragraph 018 and paragraph 019).

(V.J) Claim 18

Claim 18 is directed to a program storage device (paragraphs 001, 006, 009, 020, 021) readable by machine (paragraph 020), tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software

component (paragraph 020). The program of instruction creates creating raw program text (P0; Step 100 in Fig. 1, paragraphs 010, 013, 014, 019) and normalized program text (P1; Step 110 in Fig. 1, paragraphs 010, 013, 014, 019) of the software component. The program of instruction compresses the raw program text and the normalized program text, to provide compressed versions (C0, C1; Step 130 in Fig. 1, paragraphs 006, 010, 015, 019). The program of instruction finds lengths of the compressed versions (Steps 140, 150 in Fig. 1, paragraphs 016, 019). The program of instructions finds a ratio of the length of the compressed raw program text to the length of the compressed normalized program text (C0/C1; Step 170 in Fig. 1, paragraphs 006, 010, 016, 017, 019) and provides a software complexity metric comprising the ratio (paragraphs 006, 017, 020).

(V.K) Claim 19

Claim 19 is directed to a program storage device (paragraphs 001, 006, 009, 020, 021) readable by machine (paragraph 020), tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component (paragraph 020). The program of instruction creates normalized program text (P1; Step 110 in Fig. 1, paragraphs 010, 013, 014, 019) and normalized unique program text (P2; Step 120 in Fig. 1, paragraphs 010, 013, 014, 019) of the software component. The program of instruction compresses the normalized program text and the normalized unique program text, to provide compressed versions (C1, C2; Step 130 in Fig. 1, paragraphs 006, 010, 015, 019). The program of instruction finds lengths of the compressed versions (Steps 150, 160 in Fig. 1, paragraphs 016, 019). The program of instructions finds a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text (C1/C2; Step 170 in Fig. 1, paragraphs 006, 010, 016, 017, 019) and provides a software complexity metric comprising the ratio (paragraphs 006, 017, 020).

(V.L) Corresponding Structure for Means of Claim 9

The structure corresponding to means for compressing each of the versions is a compression application such as the open source bzip2 program and equivalents (paragraph 015). The structure corresponding to means for comparing the lengths of the compressed versions is a divider such as a microprocessor and equivalents (paragraph 018). A display other output device or equivalent provides a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraphs 006, 017, 020).

(V.M) Corresponding Structure for Means of Claim 10

The structure corresponding to means for compressing the raw program text and the normalized program text is a compression application such as the open source bzip2 program and equivalents (paragraph 015). The structure corresponding to means for finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text is a divider such as a microprocessor and equivalents (paragraph 018).

(V.N) Corresponding Structure for Means of Claim 11

The structure corresponding to means for compressing normalized program text and the normalized unique program text is a compression application such as the open source bzip2 program and equivalents (paragraph 015). The structure corresponding to means for finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text is a divider such as a microprocessor and equivalents (paragraph 018). A display other output device or equivalent provides a software complexity metric comprising a comparison of the lengths of the compressed versions (paragraphs 006, 017, 020).

(VI). Grounds of Rejection to be reviewed on appeal

Each of claims 9-19 is rejected under 35 U.S.C. 102 as being anticipated by U.S. Patent Number 6,496,974 to Sliger et al. (hereafter Sliger).

The questions for appeal are whether or not each of claims 9-19 is anticipated by Sliger under 35 U.S.C. 102.

(VII). Argument

(VII.A) Principles of Law Relating to Anticipation

The Examiner must make a prima facia case of anticipation. "A person shall be entitled to a patent unless. . . (b) the invention was patented or described in a printed publication in this or a foreign country . . . more than one year prior to the date of the application for patent in the United States." 35 U.S.C. 102. It is settled law that each element of a claim must be expressly or inherently described in a single prior art reference to find the claim anticipated by the reference. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." Verdegaal Bros. v. Union Oil Co. of California, 814 F.3d 63, 631, 2USPQ2d 1051,1053 (Fed. Cir. 1987), cert. denied, 484 U.S. 827 (1987). Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient." In re Robertson, 169 F.3d 743, 745, 49 USPQ2d 1949, 1951 (Fed. Cir. 1999)(citations and internal quotation marks omitted). The Examiner has failed to make a prima facia case of anticipation, because the claims on appeal include various elements that are not expressly or implicitly described in the reference cited (i.e., Grimm).

(VII.B) Rejection of Claim 9 under 35 USC 102 over US Pat. No. 6,496,974 (Sliger)

Applicants have contended that claim 9 is allowable because it includes features that are neither disclosed nor suggested by Sliger or any other references, either individually or in combination.

(VIII.B.1) "logic for determining a plurality of versions of the software component"

Applicant respectfully contends that claim 9 is allowable because in includes a feature that is neither disclosed nor suggested by Slinger or any other reference cited, namely "logic for determining a plurality of versions of the software component." In the present application, the logic determines two or more versions of the same software component. The versions may be for example: raw program text P0, normalized program text P1, and/or normalized unique program text P2.

Slinger is directed to updating computer software using patch files, generation of patch files, and normalization of files to which such patches are applied. Thus, Sliger is not related to providing software metrics. Moreover, Slinger does not disclose or suggest determining a plurality of versions of a software component. The Examiner previously concluded in error that this feature is disclosed by Slinger at Col. 7 lines 66-67 and col. 8 lines 1-9. Applicants respectfully disagree. The cited paragraph discusses how an older version (i.e. release) of software is displayed in a history window; characters from a new file are added to the window to supplement the compression dictionary as they are processed. Slinger does not determine a plurality of versions of a single software component. To the extent that Slinger refers to an older version and a new version, they are referring to different releases of a software title. Release 2.04 and release 3.02 of a software title are not two versions of the same software, but rather two different software components with a common title. Moreover, Slinger does not determine two versions of a software component, but rather uses a patch program to convert an older release to a new release.

Applicant then concluded in error that Sliger discloses this feature at column 12, lines 47-54. Applicant again respectfully disagrees. The cited paragraph discusses that

the patch method can also be used to replace an newer release of a software application with an older release. Again, the software releases (referred to by Sliger as versions) are not the same software, but rather have changes or differences as stated by Sliger at Column 7, lines 32-34. As clearly pointed out in the present Application the plurality of versions are different expressions of the same software component such as raw program text, normalized program text and normalized unique program text.

The mere fact that compression and normalizing are discussed in Sliger is irrelevant. Applicant is not claiming compression or normalization. The present feature is directed to logic that determines a plurality of versions of a software component, not logic that adds characters from a new release to a compression dictionary. Nor is the present feature directed to the normalization process. Sliger neither discloses nor suggests determining a plurality of versions of a single software component.

The Examiner has failed to make a prima facia case that the element "logic for determining a plurality of versions of the software component" is anticipated by Sliger.

Moreover, since this feature is not disclosed or suggested by the cited art, nor is it an obvious variation of any known reference, no rejection under 35 USC 103 would be proper. Also, software metrics have been used in software development for a long time, and the failure of others to provide this feature is evidence of its nonobviousness.

***(VII.B.2)* "means for comparing the lengths of the compressed versions"**

Applicant respectfully contends that claim 9 is also allowable because in includes another feature that is neither disclosed nor suggested by Slinger or any other reference cited, namely "means for comparing the lengths of the compressed versions". As clearly pointed out in the present application, a comparison of the lengths of compressed files of different versions of a software component can be used to provide a highly accurate measure of the true complexity of a software component by revealing characteristics that would otherwise be obscured. For example, dependencies between strings within a

software component can be taken into account by comparing lengths of compressed raw and normalize versions of the software component. Slinger neither discloses not suggests comparing the lengths of compressed versions. The Examiner had concluded in error that this feature is disclosed by Sliger at col. 3 lines 6-20. Applicants respectfully disagree. The cited passage discusses simultaneously matching an old release and a new release to generate a patch file and compressing the patch file. Thus, slinger is not comparing lengths of different compressed versions, but rather comparing code to identify differences to generate a patch to convert the old release to the new release. The Examiner then concluded in error that this feature is disclosed by Sliger at column 7, lines 19-35. Applicants again respectfully disagree. The cited paragraph discusses finding differences between an old and a new release of a software application. This is not comparing the lengths of compressed files, but rather identifying differences between compressed files.

The Examiner has failed to make a prima facia case that the element "means for comparing the lengths of the compressed versions" is anticipated by Sliger.

Moreover, since this feature is not disclosed or suggested by the cited art, nor is it an obvious variation of any known reference, no rejection under 35 USC 103 would be proper. Also, software metrics have been used in software development for a long time, and the failure of others to provide this feature is evidence of its nonobviousness.

(VII.B.3) "means for providing a software complexity metric comprising a comparison of the lengths of the compressed versions."

Applicant respectfully contends that claim 9 is also allowable because in includes another feature that is neither disclosed nor suggested by Slinger or any other reference cited, namely "means for providing a software complexity metric comprising a comparison of the lengths of the compressed versions." Slinger neither discloses nor suggests providing a complexity metric. The Examiner had concluded in error that this feature is disclosed by Slinger at col. 10 lines 10-17. Applicants respectfully disagree. The cited paragraph discusses comparing old and new releases, referred to as versions.

This is done to generate a patch file. Thus, the comparison is of code itself, not lengths of compressed code. Moreover, the cited paragraph, and every other paragraph of Slinger fails to disclose or suggest a complexity metric. In sharp contrast, claim 9 claims a metric that compares lengths of different versions of a software component after compression of the versions. These comparisons can be used to accurately access features of the software component such as redundancy of the implementation of the software component and propagation of defects as stated in paragraph {017} of the specification.

The Examiner then concluded in error that this feature is disclosed by Sliger at Col. 3, lines 5-18. Applicants again respectfully disagree. The cited paragraph discusses a process of comparing old and new releases of an application during compression to produce a compressed output (patch) by which the latter can be generated from the former. Thus, Sliger provides a patch, which is substantially different from a complexity metric. A complexity metric in contrast to a patch is a measure of complexity of a software component. For example a complexity metric might be a ratio that is an accurate factor for estimating the time needed to write, debug or test the software component.

The Examiner has failed to make a prima facia case that the element "means for comparing the lengths of the compressed versions" is anticipated by Sliger.

Moreover, since this feature is not disclosed or suggested by the cited art, nor is it an obvious variation of any known reference, no rejection under 35 USC 103 would be proper. Also, software metrics have been used in software development for a long time, and the failure of others to provide this feature is evidence of its nonobviousness.

(VII.C) Rejection of Claim 10 under 35 USC 102 over US Pat. No. 6,496,974 (Sliger)

Applicants have contended that claim 10 is allowable because it includes features that are neither disclosed nor suggested by Sliger or any other references, either individually or in combination.

(VII.C.1) features previously discussed under claim 9

Claim 10 include the features of claim 9 presented above and Applicants respectfully contend that it is allowable for the reasons presented above.

(VII.C.2) "finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text"

Applicants respectfully contends that claim 10 also allowable because it includes an additional feature that is neither disclosed nor suggested by Slinger or any other reference cited, namely that the comparing means finds a ratio of the length of the compressed version of the raw program text (C0) to the length of the compressed version of normalized program text (C1). This metric is proportional to the redundancy of the software component and is a useful metric for this attribute {017}. Sliger neither discloses nor suggests this feature.

The Examiner has concluded in error that this feature is disclosed by Sliger at col. 4, lines 9-25. Applicants respectfully disagree. The cited paragraph discusses creating a patch using a Markov model compressor. The compressor is pre-initialized with the old data file. The compressor creates and uses a probability profile when a new file is applied to the pre-initialized compressor producing a compact output file (patch). Neither the probably profile (likelihood of encountering a symbol X) nor the output file (patch) is in any way equivalent to or suggestive of a ratio of lengths of compressed raw program text to compressed normalized program text.

The Examiner has failed to make a prima facia case that the element "finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text" is anticipated by Sliger.

(VII.D) Rejection of Claim 11 under 35 USC 102 over US Pat. No. 6,496,974 (Sliger)

Applicants have contended that claim 11 is allowable because it includes features that are neither disclosed nor suggested by Sliger or any other references, either individually or in combination.

(VII.D.1) features previously discussed under claim 9

Claim 10 include the features of claim 9 presented above and Applicants respectfully contend that it is allowable for the reasons presented above.

(VII.D.2) "finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text"

Applicants respectfully contends that claim 10 also allowable because it includes an additional feature that is neither disclosed nor suggested by Slinger or any other reference cited, namely that the comparing means finds a ratio of the length of the compressed version of the normalized program text (C1) to the length of the compressed version of normalized unique program text (C2). This metric is proportional to the propagation of defects for the software component and is a useful metric for this attribute {017}. Sliger neither discloses nor suggests this feature.

The Examiner has concluded in error that this feature is disclosed by Sliger at col. 4, lines 9-25. Applicants respectfully disagree. The cited paragraph discusses creating a patch using a Markov model compressor. The compressor is pre-initialized with the old data file. The compressor creates and uses a probability profile when a new file is applied to the pre-initialized compressor producing a compact output file (patch). Neither the probably profile (likelihood of encountering a symbol X) nor the output file

(patch) is in any way equivalent to or suggestive of a ratio of lengths of compressed raw program text to compressed normalized program text.

The Examiner has failed to make a prima facia case that the element "finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text" is anticipated by Sliger.

Moreover, since this feature is not disclosed or suggested by the cited art, nor is it an obvious variation of any known reference, no rejection under 35 USC 103 would be proper. Also, software metrics have been used in software development for a long time, and the failure of others to provide this feature is evidence of its nonobviousness.

(VII.E) Rejection of Claims 12-19 under 35 USC 102 over US Pat. No. 6,496,974 (Sliger)

(VII.D.1) features previously discussed under claim 9

Claims 12-19 are directed to a program storage device having a program of instructions to perform the functions of the apparatus of claims 9-11. Applicant respectfully contends that claims 12-19 are allowable for the reasons that claims 9-11 are allowable.

Sincerely,

Steven E. Bach
Attorney for Applicants
Reg. No. 46,530

(VIII) Claims Appendix

Listing of Claims:

1. (Canceled)

2. (Canceled)

3. (Canceled)

4. (Canceled)

5. (Canceled)

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. (previously presented) Apparatus for determining complexity of a software component, comprising:

logic for determining a plurality of versions of the software component and for finding lengths of compressed versions of the plurality of versions of the software;

means for compressing each of the versions, to provide the compressed versions;

means for comparing the lengths of the compressed versions; and

means for providing a software complexity metric comprising a comparison of the lengths of the compressed versions.

10. (previously presented) Apparatus for determining complexity of a software component, comprising:

logic for creating raw program text and normalized program text of the software component and for finding lengths of compressed raw program text and compressed normalized program text;

means for compressing the raw program text and the normalized program text to provide the compressed raw program text and the compressed normalized program text, respectively; and

means for finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text; and

means for providing a complexity metric comprising the ratio.

11. (currently amended) Apparatus for determining complexity of a software component, comprising:

logic for creating normalized program text and normalized unique program text of the software component and for finding lengths of compressed normalized program text and compressed normalized unique program text;

means for compressing the normalized program text and the normalized unique program text to provide the compressed normalized program text and the compressed normalized unique program text, respectively; and

means for finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text; and

means for providing a complexity metric comprising the ratio.

12. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component, said method steps comprising:

creating a plurality of versions of the software component;

compressing each of the versions, to provide compressed versions;

finding lengths of the compressed versions;

comparing the lengths of the compressed versions; and

providing a software complexity metric comprising a comparison of the lengths of the compressed versions.

13. (previously presented) The program storage device of claim 12, wherein the plurality of versions comprises raw program text.

14. (previously presented) The program storage device of claim 12, wherein the plurality of versions comprises normalized program text.

15. (previously presented) The program storage device of claim 12, wherein the plurality of versions comprises normalized unique program text.

16. (previously presented) The program storage device of claim 12, wherein the step of comparing comprises a step of finding a ratio using the length of the compressed version of raw program text and the length of the compressed version of normalized program text.

17. (previously presented) The program storage device of claim 12, wherein the step of comparing comprises a step of finding a ratio using the length of the compressed version

of normalized program text and the length of the compressed version of normalized unique program text.

18. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component, said method steps comprising:

creating raw program text and normalized program text of the software component;

compressing the raw program text and the normalized program text to provide compressed raw program text and compressed normalized program text, respectively;

finding the length of the compressed raw program text and the length of the compressed normalized program text;

finding a ratio of the length of the compressed raw program text to the length of the compressed normalized program text; and

providing a software complexity metric comprising the ratio.

19. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by machine to perform method steps for determining complexity of a software component, said method steps comprising:

creating normalized program text and normalized unique program text of the software component;

compressing the normalized program text and the normalized unique program text to provide compressed normalized program text and compressed normalized unique program text, respectively;

finding the length of the compressed normalized program text and the length of the compressed normalized unique program text;

finding a ratio of the length of the compressed normalized program text to the length of the compressed normalized unique program text; and

providing a software complexity metric comprising the ratio.

(IX). Evidence appendix

There is no extrinsic evidence provided.

(X). Related proceedings appendix

There are no related proceedings.